

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

|                                  |   |                                  |
|----------------------------------|---|----------------------------------|
| In re Application of:            | § |                                  |
| Jason Robert Almeida             | § | Group Art Unit: 2136             |
|                                  | § |                                  |
| Serial No.: 10/044,432           | § | Examiner: Cervetti, David G.     |
|                                  | § |                                  |
| Filed: January 11, 2002          | § | Atty Docket No.: RPS920010091US1 |
|                                  | § |                                  |
| Title: Method And System For     | § | Customer No.: 34533              |
| Programming A Non-Volatile       | § |                                  |
| Device In A Data Processing Sys. | § | Confirmation No.: 8540           |

**Mail Stop: Appeal Brief-Patents**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, Virginia 22313-1450

**REPLY BRIEF TO EXAMINER'S ANSWER OF AUGUST 2, 2006**

**Honorable Commissioner:**

This is a Reply Brief to the Examiner's Answer of August 2, 2006, pursuant to 37 CFR § 41.41. The Appeal Brief was filed pursuant to 37 CFR § 41.37 on June 12, 2006, in response to the Final Office Action of October 12, 2005, and pursuant to the Notice of Appeal filed March 12, 2006.

**REAL PARTY IN INTEREST**

The real party in interest is the patent assignee, International Business Machines Corporation ("IBM"), a New York corporation having a place of business at Armonk, New York 10504.

## RELATED APPEALS AND INTERFERENCES

There are no related appeals no interferences known to Appellant which will directly affect, be directly affected by, or have a bearing on the Board's decision in this appeal.

## STATUS OF CLAIMS

Claims 1-24 are pending in this application. Claims 1-24 stand rejected under the Final Office Action. More particularly: Claims 1-2, 4-5, 9-10, 12-13, 17-18, and 20-21 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Bright *et al.* (U.S. Patent No. 6,141,756), hereinafter "Bright." Claims 3, 11, and 19 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bright in view of Hughes (U.S. Patent No. 5,968,174), hereinafter "Hughes." Claims 6-7, 14-15, and 22-23 stand rejected under 35 USC § 103(a) as being unpatentable over Bright in view of Cuccia *et al.* (U.S. Patent No. 6,151,676), hereinafter "Cuccia." Claims 8, 16, and 24 also stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bright, and further in view of Cuccia. Appellant lists the rejection of these three claims separately from the rejection of claim 6-7, 14-15, and 22-23 despite the common references to maintain consistency with the form in which the Office Action identifies the grounds of rejection.

## STATUS OF AMENDMENTS

No amendments were submitted after final rejection. The claims as currently presented are included in the Appendix of Claims that accompanies this Reply Brief.

## SUMMARY OF CLAIMED SUBJECT MATTER

Applicants provide the following concise summary of the invention according to 37 CFR 1.192(c)(5):

Independent claim 1 recites computer executable instructions (computer code means), stored on a computer readable medium for programming a non-volatile storage element (105) of a data processing system (100) (see page 5, lines 1-4). The claim includes instructions for encrypting (304) a digital signature using a first encryption key [page 8,

lines 8-9] and passing the encrypted signature (203) to a kernel routine (210) [page 8, lines 9-11]. Upon successfully decrypting (308) the signature (203), the kernel routine (210) transitions (311) system (100) to a real-mode state before calling a real mode flashing routine (204) to flash program (312) the non-volatile storage element (105) [page 6, lines 3-20], [page 8, lines 8-18].

Independent claim 9 recites a data processing system (100) including at least one processor (102), memory (106), and input means connected to a common bus (104, 110), [page 3, line 30 through page 4, line 19]. The system memory (106) contains at least a portion of a sequence of computer executable instructions for programming a non-volatile storage element (105) of the system (100). The instructions parallel the instructions recited in claim 1. Specifically, instructions for encrypting (304) a digital signature using a first encryption key [page 8, lines 8-9] and passing the encrypted signature (203) to a kernel routine (210) [page 8, lines 9-11]. Upon successfully decrypting (308) the signature (203), the kernel routine (210) transitions (311) system (100) to a real-mode state before calling a real mode flashing routine (204) to flash program (312) the non-volatile storage element (105) [page 6, lines 3-20], [page 8, lines 8-18].

Independent claim 17 recited a method (300) for programming a non-volatile storage element (105) in a data processing system. The method includes elements that parallel the code elements of independent claim 1. Specifically, the method (300) includes encrypting (304) a digital signature using a first encryption key [page 8, lines 8-9] and passing the encrypted signature (203) to a kernel routine (210) [page 8, lines 9-11]. Upon successfully decrypting (308) the signature (203), the kernel routine (210) transitions (311) system (100) to a real-mode state before calling a real mode flashing routine (204) to flash program (312) the non-volatile storage element (105) [page 6, lines 3-20], [page 8, lines 8-18].

Claims 1-16 all recite “computer code means,” stored on a computer readable medium, for encrypting, passing, transitioning, programming, and so forth. Appellant is cognizant

of the requirement under 37 CFR 41.37(c)(1)(v) to identify every means plus function and step plus function and to set forth the structure, material, or acts described in the specification as corresponding to each claimed function. To the extent the requirement is applicable to Beauregard claims, Appellant identify and set forth as follows:

Claim 1, computer code means for encrypting a digital signature using a first encryption key. For the acts described in the specification as corresponding to this element, see 304 of FIG 3 and accompanying text at page 8, lines 8-9.

Claim 1, computer code means for passing the encrypted signature to a kernel routine. For the acts described in the specification as corresponding to this element, text at page 8, lines 9-11.

Claim 1, computer code means, responsive to successfully decrypting the encrypted signature using a second encryption key, for transitioning the data processing system from a protected-mode to a real-mode. For the acts described in the specification as corresponding to these elements, see elements 308, 310, and 311 of FIG 3 and accompanying text at page 8, lines 11-15.

Claim 1, real-mode computer code means for flash programming the non-volatile storage element. For the acts described in the specification as corresponding to this element, see element 312 of FIG 3 and accompanying text at page 8, lines 15-16.

Claim 9, computer code means for encrypting a digital signature using a first encryption key. For the acts described in the specification as corresponding to this element, see 304 of FIG 3 and accompanying text at page 8, lines 8-9.

Claim 9, computer code means for passing the encrypted signature to a kernel routine. For the acts described in the specification as corresponding to this element, text at page 8, lines 9-11.

Claim 9, computer code means, responsive to successfully decrypting the encrypted signature using a second encryption key, for transitioning the data processing system from a protected-mode to a real-mode. For the acts described in the specification as corresponding to these elements, see elements 308, 310, and 311 of FIG 3 and accompanying text at page 8, lines 11-15.

Claim 9, real-mode computer code means for flash programming the non-volatile storage element. For the acts described in the specification as corresponding to this element, see element 312 of FIG 3 and accompanying text at page 8, lines 15-16.

### **NEW GROUND OF REJECTION**

The new question presented is:

1. Whether additional references in Bright cited in the Examiner's Answer cure the deficiencies of Bright to disclose each and every element of claims 1, 9, and 17 within the meaning of 35 U.S.C. § 102?

### **ARGUMENT**

Appellant presents the following arguments pursuant to 37 CFR § 41.37(c)(1)(vii) regarding the new ground of rejections in the present case.

**NEW GROUND OF REJECTION: WHETHER ADDITIONAL REFERENCES IN  
BRIGHT CITED IN THE EXAMINER'S ANSWER CURE THE DEFICIENCIES  
OF BRIGHT TO DISCLOSE EACH AND EVERY ELEMENT OF CLAIMS  
1, 9, AND 17 WITHIN THE MEANING OF 35 U.S.C. § 102**

Claims 1, 9, and 17 stand rejected under 35 U.S.C § 102 as being anticipated by Bright *et al.* (U.S. Patent No. 6,141,756) in the Final Office Action of October 12, 2005. To anticipate claims 1, 9, and 17 under 35 U.S.C. § 102, two basic requirements must be met. The first requirement of anticipation is that Bright must disclose each and every element as set forth in Appellant's claims. The second requirement of anticipation is that Bright must enable Appellant's claims. The Appeal Brief dated June 12, 2006 presented

arguments that Bright does not anticipate Appellant's claims. In response to the Appeal Brief, the Examiner's Answer cites additional references in Bright in an attempt to demonstrate that Bright does anticipate the Appellant's claims within the meaning of 35 U.S.C. § 102. As explained in detail below, the additional references cited in Bright by the Examiner's Answer do not meet the requirements of anticipation, and the rejection should be withdrawn.

Bright Does Not Disclose Each And  
Every Element Of Claim 1

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Bright generally discloses reading a program into a processor and does not disclose each and every element of Applicants' claim 1. Independent claim 1 of the present application claims:

1. A computer program product comprising processor executable instructions for programming a non-volatile storage element in a data processing system, the instructions being stored on a computer readable medium, comprising:

computer code means for encrypting a digital signature using a first encryption key;

computer code means for passing the encrypted signature to a kernel routine;

computer code means, responsive to successfully decrypting the encrypted signature using a second encryption key, for transitioning the data processing system from a protected-mode to a real-mode; and

real-mode computer code means for flash programming the non-volatile storage element.

Bright Does Not Disclose Responsive To Successfully Decrypting The  
Encrypted Signature Using A Second Encryption Key, For Transitioning  
The Data Processing System From A Protected-Mode To A Real-Mode

The third element of claim 1 claims “computer code means, responsive to successfully decrypting the encrypted signature using a second encryption key, for transitioning the data processing system from a protected-mode to a real-mode...” The Final Office Action relies on Bright at column 4, lines 14-32, in an effort to show that Bright discloses the third element of claim 1. In response to the Final Office Action, the Appeal Brief explains that column 4, lines 14-32, of Bright merely discloses decrypting an encrypted program provided by an external device. The Appeal Brief further explains that Bright at column 4, lines 14-32, neither explicitly nor inherently discloses anything regarding transitioning a system from protected-mode to a real-mode. The Appeal Brief points out that the terms ‘protected mode’ and ‘real mode’ refer to two different operating modes of a system. Real-mode refers to a single-tasking-mode in which a program executing on the computer system has full and direct access to the computer’s memory and peripherals. Protected-mode is an operating mode that lets software utilize memory beyond 16-bit addressing and creates a protection scheme enabling multiple programs to share a common set of computer resources without conflicting with each other. Bright never even once mentions the terms ‘protected-mode’ or ‘real-mode.’ The Appeal Brief notes that the Appellant is not surprised with Bright’s failure to mention these terms because Bright is not concerned with transitioning a system from protected mode to real mode as part of a flash programming routine. Instead, Bright is concerned with executing an encrypted program downloaded from an external device. As such, Appellant argued in the Appeal Brief that Bright does not disclose each and every element of claim 1 and the rejection should be withdrawn.

The Examiner’s Answer at page 11 responds to the arguments in the Appeal Brief regarding the third element of claim 1 stating:

Regarding Appellant's assertion that Bright does not disclose transitioning from protected to real modes or a kernel routine, Examiner respectfully disagrees with the Appellant.

Bright discloses two modes, a mode where sensitive processing occurs (bootstrap mode), which corresponds to Applicant's "protected mode", and transition to execution of program upon successful decryption/validation/authentication, which is inherent, real mode (column 1, lines 10-45).

Examiner further refers to figure 3, steps 301-315 which correspond to "protected mode", perform decryption, and at step 317 executes the program which is executing in a non-secure mode, which corresponds to Applicant's "real mode", an inherency of the system (column 3, line 58, to column 5, line 12). ...

That is, the Examiner's Answer argues that the bootstrap mode of Bright discloses the protected-mode as claimed in the present application and that executing a program outside of Bright's bootstrap mode inherently discloses the real-mode as claimed in the present application. In support of such arguments, the Examiner's Answer now cites Bright at column 1, lines 10-45, and steps 301-317 of Figure 3. Applicants respectfully note that neither reference in Bright supports the Examiner's position that Bright discloses the protected-mode as claimed in the present application and that executing a program outside of Bright's bootstrap mode inherently discloses the real-mode as claimed in the present application.

Regarding Bright at column 1, lines 10-45, Applicants respectfully note in response that Bright at column 1, lines 10-45, discloses:

Processors, such as microprocessors, digital signal processors, programmable logic arrays (PLAs), field programmable gate arrays (FPGAs) micro controllers, and microcomputers, are well known. Such devices may include on-board RAM (Random Access Memory), ROM (Read Only Memory), EPROM, timers, I/O ports, and serial ports.

Processors often have a bootstrap mode, also known programming, emulation (debug), or test mode, which entails downloading a bootstrap program (or other data) from an external source, which program is executed by the processor to provide a desired function, which functions are numerous. Devices such as PLAs and other reconfigurable hardware



devices also have a bootstrap mode that serves to provide internal hardware configurations from an external device that contains instructions or blueprints for configuration of the device.

Because the program being downloaded in bootstrap mode comes from a source external to the processor, there are potential security risks associated with downloading an external program. Today, security for microprocessors is designed to prevent further reading in and reading out of data using fuses or fusible links that are severed once the program is entered into the microprocessor. Such technology may be found in the PIC.TM. chip available from Microchip, Inc. Such a solution, however, does not prevent tampering with the external source of information nor does it prevent undesirable programs from entering the processor, and further prevents the device from being programmed at a later time by an authorized programmer, thereby limiting the flexibility of the device.

Accordingly, there is a need for a processor that has a more secure method of downloading a bootstrap program that allows for multiple bootstrap programming of a single device.

That is, Bright at column 1, lines 10-45, discloses a bootstrap mode that entails downloading a bootstrap program or other data on a processor from an external source and executing the program by the processor to provide a desired function. Bright further states that more secure methods of downloading bootstrap programs are needed to prevent malicious code being executed by a processor, and the remainder of Bright goes on to discuss how to implement such a secure bootstrap mode. Regardless of whether a bootstrap mode is secure or insecure, however, Bright's bootstrap mode has nothing whatsoever to do with a protected-mode as claimed in the present application. As mentioned above, the protected-mode is an operating mode that lets software utilize memory beyond 16-bit addressing and creates a protection scheme enabling multiple programs to share a common set of computer resources without conflicting with each other. In contrast, nowhere in column 1, lines 10-45, does Bright disclose that a bootstrap mode allows software to access memory beyond 16-bit addressing or that creates a protection scheme enabling multiple programs to share a common set of computer resources without conflicting with each other. In fact, Bright at column 1, lines 10-45, never even mentions anything related to address space and only describes one program—the bootstrap program—being executed on the processor. The bootstrap mode

of Bright, therefore, does not disclose the protected mode as claimed in the present application. Because Bright does not disclose each and every element and limitation of the Appellant's claims, the rejections should be withdrawn.

Furthermore, even if Bright's bootstrap mode disclosed a protected-mode as claimed in the present invention, which it does not, Bright at column 1, lines 10-45, still does not disclose transitioning a data processing system from a protected-mode to a real-mode as claimed in the present application. Nowhere in Bright at column 1, lines 10-45, is the system transitioned from a bootstrap mode to a real-mode as claimed in the present application. In fact, Bright at column 1, lines 10-45, does not mention transitioning at all. Because Bright does not disclose each and every element and limitation of the Appellant's claims, the rejections should be withdrawn.

By attempting to equate the bootstrap mode of Bright with the protected-mode as claimed in the present application, Appellants respectfully point out that the Examiner exceeds the boundaries of the requirement during prosecution to interpret the claims as broadly as their terms reasonably allow. *In re American Academy of Science Tech Center*, 367 F.3d 1359, 1369, 70 USPQ2d 1827, 1834 (Fed. Cir. 2004). When interpreting the claims as broadly as their terms reasonably allow, the words of the claims must be given their plain meaning unless Applicants have provided a clear definition in the specification. *In re Zletz*, 893 F.2d 319, 321, 13 USPQ2d 1320, 1322 (Fed. Cir. 1989); *Chef America, Inc. v. Lamb-Weston, Inc.*, 358 F.3d 1371, 1372, 69 USPQ2d 1857 (Fed. Cir. 2004). In the present case, the Examiner attempts to broadly interpret Appellant's claim term 'protected-mode' to encompass Bright's bootstrap mode that protects a processor from executing malicious computer code. Such an interpretation, however, conflicts with the clear definition of the term 'protected-mode' provided by the Appellants in the original specification at the paragraph beginning on page 2, line 4, stating:

Protected-mode is an operating-mode (introduced with the 80286 microprocessor) that lets software use memory beyond 16-bit addressing (640 KB). Protected-mode also creates a protection scheme enabling

multiple programs to share a common set of computer resources (such as system memory) without conflicting with each other.

Clearly, the Examiner's interpretation of the claim term 'protected-mode' to encompass protecting a processor from executing malicious computer code conflicts with the clear definition provided by the Appellant. The Examiner's interpretation of 'protected-mode,' therefore, is incorrect, and the rejections based on the Examiner's incorrect interpretation should be withdrawn.

In addition to Bright at column 1, lines 10-45, readers will recall that the Examiner's answer also references Bright at Figure 3, steps 301-317, to support the assertion that the bootstrap mode of Bright discloses the protected-mode as claimed in the present invention. Appellant respectfully note in response, however, that Bright at Figure 3, steps 301-317, merely discloses a flowchart of a method of decrypting and authenticating a bootstrap program. The flowchart of Figure 3 illustrates the steps of entering a bootstrap mode, reading a program from an external device, decrypting the program if necessary, authenticating the program if necessary, and executing the program. Because none of the steps of illustrated in Figure 3 of Bright having anything to do with an operating mode that lets software utilize memory beyond 16-bit addressing or creating a protection scheme enabling multiple programs to share a common set of computer resources without conflicting with each other, Figure 3 of Bright does not disclose a protected-mode as claimed in the present application. Because Bright does not disclose each and every element and limitation of the Appellant's claims, the rejections should be withdrawn.

Turning now, the Appellant addresses the assertion in the Examiner's Answer that Bright at column 1, lines 10-45, and Figure 3, steps 301-317, inherently discloses the real-mode as claimed in the present application. Regarding the Examiner's invocation of the theory of inherency, Appellants respectfully point out that the rejection is not accompanied by the required analysis to support a rejection relying on inherency. Merely reciting the word 'inherently' or 'inherent' is an insufficient basis for a rejection on the theory of inherency. "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the

allegedly inherent characteristic necessarily flows from the teachings of the applied prior art.” *Manual of Patent Examination Procedure* § 2112 (citing *Ex parte Levy*, 17 USPQ2d 1461, 1464 (Bd. Pat. App. & Inter. 1990)). The Examiner’s Answer relies on the theory of inherency to asserts that Bright at column 1, lines 10-45, and Figure 3, steps 301-317, discloses the real-mode as claimed in the present application because the decrypted bootstrap program of Bright is executed in a non-secure mode. The Examiner’s Answer, however, does not demonstrate that executing a decrypted bootstrap program in a non-secure mode necessarily results in transitioning a data processing system from a protected-mode to a real-mode as claimed in the present application. The Examiner’s Answer, therefore, has not demonstrated the required analysis to support a rejection relying on inherency. For this reason alone, the rejections relying on inherency should be withdrawn.

Moreover, the inherency relied upon by the Examiner does not exist. As mentioned above, the real-mode as claimed in the present application is a single-tasking-mode in which a program executing on the computer system has full and direct access to the computer’s memory and peripherals. Bright’s disclosure of executing a decrypted program in a non-secure mode does not result in transitioning a data processing system from a protected-mode to a real-mode as claimed in the present application because the secure bootstrap mode of Bright used to decrypt the bootstrap program is not the protected-mode as claimed in the present application for all the reasons discussed above. Even if Bright’s bootstrap mode discloses the protected-mode, which it does not, nothing about executing a decrypted program in a non-secure mode of Bright necessarily results in transitioning from a protected-mode to a single-tasking-mode in which a program executing on the computer system has full and direct access to the computer’s memory and peripherals. Because the inherency relied upon by the Examiner does not exist, the rejections relying on inherency should be withdrawn.

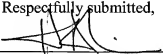
Conclusion To Appellant's Argument

Claims 1, 9, and 17 stand rejected under 35 U.S.C § 102 as being anticipated by Bright *et al.* (U.S. Patent No. 6,141,756). For the reasons explained above, Bright does not disclose each and every element and limitation of independent claim 1 and, therefore, does not anticipate independent claim 1. Accordingly, Appellants submit that independent claim 1 is patentable and should be allowed. Independent claims 9 and 17 claim system aspects and method aspects of the computer program product claimed in independent claim 1. Independent claims 9 and 17, therefore, are patentable and should be allowed for the same reasons that independent claim 1 is patentable and should be allowed.

In view of the forgoing arguments, Appellant submits that the anticipation rejections of independent claims 1, 9, and 17 are improper, and Appellant respectfully requests the Board to reverse the rejection of these claims and remand the case to the Examiner with an order to allow the claims or issue a properly founded rejection.

The Commissioner is hereby authorized to charge or credit Deposit Account No. 09-0447 for any fees required or overpaid.

Date: October 2, 2006

Respectfully submitted,  
By:   
H. Artoush Ohanian  
Reg. No. 46,022  
Biggers & Ohanian, LLP  
P.O. Box 1469  
Austin, Texas 78767-1469  
Tel. (512) 472-9881  
Fax (512) 472-9887  
ATTORNEY FOR APPELLANTS

## APPENDIX OF CLAIMS ON APPEAL

### CLAIMS

What is claimed is:

1. A computer program product comprising processor executable instructions for programming a non-volatile storage element in a data processing system, the instructions being stored on a computer readable medium, comprising:

computer code means for encrypting a digital signature using a first encryption key;

computer code means for passing the encrypted signature to a kernel routine;

computer code means, responsive to successfully decrypting the encrypted signature using a second encryption key, for transitioning the data processing system from a protected-mode to a real-mode; and

real-mode computer code means for flash programming the non-volatile storage element.

2. The computer program product of claim 1, wherein the code means for encrypting the digital signature is non-privileged code.
3. The computer program product of claim 2, wherein the code means for passing the encrypted signature to the kernel routine comprises code means for executing a system call from the non-privileged code and passing the signature as a parameter of the system call.

4. The computer program product of claim 1, wherein the first encryption key is a private key and the second encryption key is a public key, wherein the public key and private key are generated from a common algorithm.
5. The computer program product of claim 1, further comprising code means for generating the digital signature, wherein the digital signature includes information that is indicative of the data processing system.
6. The computer program product of claim 5, wherein the digital signature is generated based at least in part upon dynamic information.
7. The computer program product of claim 6, wherein the digital signature is generated at least in part based further upon information including a corresponding hostname and process ID.
8. The computer program product of claim 1, further comprising code means for generating a random number as the digital signature.
9. A data processing system including at least one processor, memory, and input means connected to a common bus, wherein the system memory contains at least a portion of a sequence of computer executable instructions for programming a non-volatile storage element of the data processing system, the instructions comprising:

computer code means for encrypting a digital signature using a first encryption key;

computer code means for passing the encrypted signature to a kernel routine;

computer code means, responsive to successfully decrypting the encrypted

signature using a second encryption key, for transitioning the data processing system from a protected-mode to a real-mode; and

real-mode computer code means for flash programming the non-volatile storage element.

10. The data processing system of claim 9, wherein the code means for encrypting the digital signature is non-privileged code.
11. The data processing system of claim 10, wherein the code means for passing the encrypted signature to the kernel routine comprises code means for executing a system call from the non-privileged code and passing the signature as a parameter of the system call.
12. The data processing system of claim 9, wherein the first encryption key is a private key and the second encryption key is a public key, wherein the public key and private key are generated from a common algorithm.
13. The data processing system of claim 9, further comprising code means for generating the digital signature, wherein the digital signature includes information that is indicative of the data processing system.
14. The data processing system of claim 13, wherein the digital signature is generated based at least in part upon dynamic information.
15. The data processing system of claim 14, wherein the digital signature is generated at least in part based further upon information including a corresponding hostname and process ID.
16. The data processing system of claim 9, further comprising code means for generating a random number as the digital signature.



17. A method of programming a non-volatile storage element in a data processing system, comprising:  
  
encrypting a digital signature using a first encryption key;  
  
passing the encrypted signature to a kernel code routine;  
  
responsive to successfully decrypting the encrypted signature using a second encryption key, transitioning the data processing system from a protected-mode to a real-mode with the kernel code routine; and  
  
flash programming the non-volatile storage element in real mode.
18. The method of claim 17, wherein encrypting the digital signature comprises encrypting the digital signature with non-privileged code.
19. The method of claim 18, wherein passing the encrypted signature to the kernel routine comprises executing a system call from the non-privileged code and passing the signature as a parameter of the system call.
20. The method of claim 17, wherein the first encryption key is a private key and the second encryption key is a public key, wherein the public key and private key are generated from a common algorithm.
21. The method of claim 17, further comprising generating the digital signature, wherein the digital signature includes information that is indicative of the data processing system.
22. The method of claim 21, wherein the digital signature is generated based at least in part upon dynamic information.

23. The method of claim 22, wherein the digital signature is generated at least in part based further upon information including a corresponding hostname and process ID.
24. The method of claim 17, further comprising code means for generating a random number as the digital signature.

**APPENDIX OF EVIDENCE**

This is an evidence appendix in accordance with 37 CFR § 41.37(c)(1)(ix).

There is in this case no evidence submitted pursuant to 37 CFR §§ 1.130, 1.131, or 1.132, nor is there in this case any other evidence entered by the examiner and relied upon by the appellants.

**RELATED PROCEEDINGS APPENDIX**

This is a related proceedings appendix in accordance with 37 CFR § 41.37(c)(1)(x).

There are no decisions rendered by a court or the Board in any proceeding identified pursuant to 37 CFR § 41.37(c)(1)(ii).